SYNTHUP
carte PCI pour la synthèse du son et le traitement de signal en temps réel

Jean-Michel Raczinski, raczinski@cemamu.asso.fr
Gérard Marino, marino@cemamu.asso.fr
Stéphane Sladek, sladek@cemamu.asso.fr
Vincent Fontalirant, fontalirant@cemamu.asso.fr

CEMAMu
CNET-B403
38-40, rue du Général Leclerc
92794 ISSY-LES-MOULINEAUX Cedex 9

RÉSUMÉ

SYNTHUP est une carte au format PCI destinée à la synthèse de son et, plus généralement, au traitement de signal en temps réel. Elle est basée sur des FPGA (Field Programmable Gate Arrays) qui lui confèrent à la fois la puissance de calcul de l'électronique câblée et la souplesse du logiciel. L'article décrit l'architecture modulaire de la carte dont l'élément de base est une cellule formée d'un FPGA pilotant deux mémoires (SDRAM $4M \times 16$). Sept cellules sont dédiées au calcul d'échantillons, la huitième se charge du contrôle de synthèse. Les cellules sont reliées entre elles à travers un réseau de FPGA qui, par ailleurs, connecte la carte avec l'extérieur. L'interface PCI est utilisée non seulement pour le transfert des données mais aussi pour la configuration de la carte (chargement des codes de synthèse). La carte est accessible soit par l'utilisation directe d'un driver de type VxD soit par l'intermédiaire de bibliothèques de classe C++. L'article présente les premiers résultats obtenus dans le cadre de l'UPIC : synthèse de 800 oscillateurs interpolés simultanément, réalisation d'un moniteur accédant aux différentes ressources de la carte. La structure des données et leur localisation (RAM du PC ou RAM de la carte) sont présentées ainsi que la façon d'y accéder à travers des tables d'adressage. L'article conclut sur les utilisations futures qui seront facilitées par la mise en place d'une boîte à outil pour le développement d'applications nouvelles.

1. Introduction

Lorsque nous avons conçu notre nouvelle station de travail, nos objectifs étaient multiples : synthèse de plusieurs centaines d'oscillateurs en temps réel ; programmabilité de l'algorithme de synthèse et du contrôle de synthèse ; extension au traitement de signal en temps réel ; découplage entre la synthèse et l'interface utilisateur ; intégration dans un micro-ordinateur du commerce ; prix raisonnable.

L'étude s'est concrétisée par la réalisation d'une carte au format PCI appelée SYNTHUP puisque destinée en priorité à la synthèse dans l'UPIC.

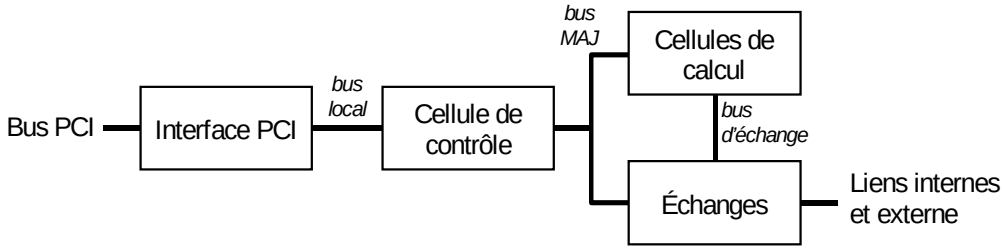
La carte est basée sur des FPGA (Field Programmable Gate Arrays) de Xilinx. Ce type de composant permet d'allier la vitesse de calcul de l'électronique câblée à la souplesse du logiciel. En effet, les fonctions réalisées par les FPGA (additions, multiplications, manipulation de bits, accès mémoire, etc.) sont programmées à l'aide d'éditeurs qui, en final, génèrent un code qui sera chargé dans le composant par l'ordinateur hôte. La carte fonctionne donc comme un coprocesseur spécialisé que l'on peut reconfigurer entièrement en un instant.

La structure des FPGA est particulièrement adaptée à l'implantation d'opérations en parallèle (logique synchrone en pipeline). On réalise par exemple plus de 20 additions 32 bits simultanées à 100 MHz dans un composant de « petite » taille (4013XLA équivalent à 13 000 portes logiques en moyenne). Une multiplication 16×16 occupe près de la moitié des ressources du même composant et fonctionne à 70 MHz.

2. Architecture

D'un point de vue fonctionnel, la carte est formée de quatre blocs :

- Une interface PCI
- Une cellule dédiée au contrôle de synthèse
- Un ensemble de cellules dédiées au calcul d'échantillons
- Un réseau d'échanges qui permet le transfert de données entre cellules et l'échange de données avec d'autres cartes



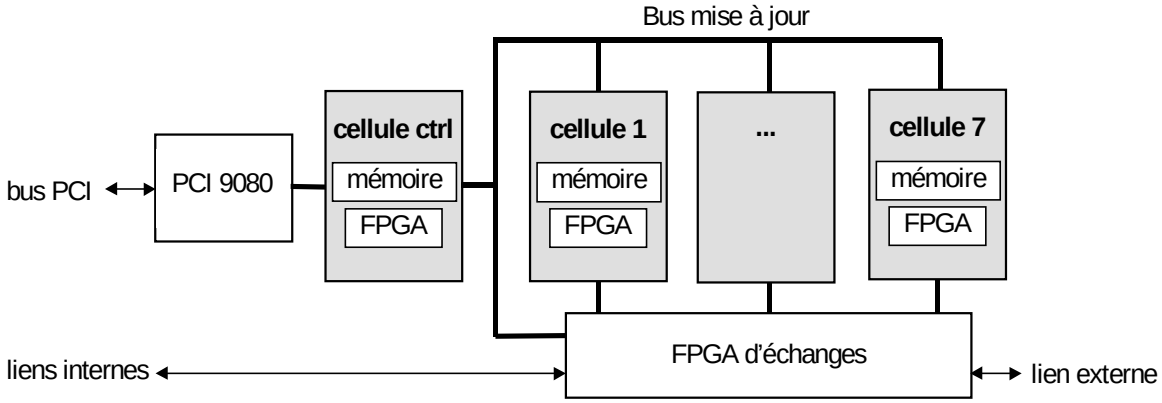
SYNTHUP : schéma fonctionnel

Nous avons multiplié le nombre de bus de communication entre blocs fonctionnels afin de tirer parti au mieux des capacités de parallélisme des FPGA.

Le bus MAJ sert à la mise à jour des paramètres de synthèse alors que le bus d'échange est utilisé à 100% par le synthétiseur pour le transfert des données de synthèse entre cellules. Ce découplage entre synthèse et contrôle de synthèse évite l'interruption de la production sonore pendant la mise à jour des paramètres.

De la même façon, nous avons séparé le bus local du bus de mise à jour afin de découpler l'interface avec l'ordinateur hôte (bus local) de la fonction « contrôle de synthèse ».

Le schéma ci-dessous montre une architecture modulaire basée sur la répétition d'une même cellule formée d'un FPGA contrôlant une mémoire de 16M octets.



SYNTHUP : cellules et FPGA d'échanges

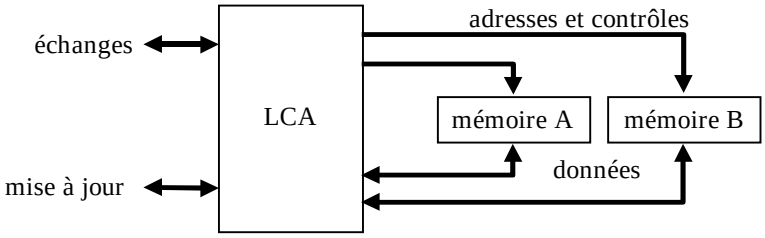
3. Interface PCI

Nous avons opté pour un composant programmable spécialisé : PCI 9080 de PLX. Parmi ses performances, nous retiendrons plus particulièrement :

- Conformité à la spécification PCI V2.1 (32 bit, 33 MHz)
- Signaux 5 V ou 3,3 V
- Plug and Play : la configuration système est stockée dans une EEPROM
- Direct Slave : le PC lit ou écrit des données en rafale dans la carte qui fonctionne en mode Target (burst memory-mapped access)
- Direct Bus Master : la carte lit ou écrit des données dans la mémoire (RAM) du PC en prenant le contrôle du bus PCI
- Deux canaux DMA indépendants avec DMA chaîné utilisé en mode scatter-gather
- Bus local configurable (8, 16 ou 32 bits, multiplexé ou non) fonctionnant à 40 MHz
- Synchronisation des données entre le bus PCI (33 MHz) et le bus local à travers huit FIFO indépendants. Transfert théorique maximum à 132M octets/seconde

L'ordinateur hôte charge le code de synthèse à travers le bus PCI. Nous avons implanté pour cela un EPLD (Electricaly Programmable Logic Device non représenté sur les figures ci-dessus) qui se charge de décoder les accès destinés au chargement de code et génère les signaux de programmation des FPGA. Cet EPLD contient par ailleurs des informations permettant d'identifier la carte (type, numéro de série, etc.).

4. Cellules



SYNTHUP : cellule

Les cellules sont formées d'un FPGA pilotant deux mémoires (SDRAM 4M x 16 conformes à la norme PC100). Comme ces mémoires sont totalement indépendantes l'une de l'autre (bus séparés), on peut effectuer deux accès simultanés à des adresses différentes (lecture et/ou écriture). C'est ainsi qu'on effectue l'interpolation de table en un seul cycle d'horloge. Il est également possible de configurer les mémoires comme une seule banque de 4M x 32. Nous avons donc un total de 128M octets sur la carte.

Les huit cellules (une cellule de contrôle et sept cellules de calcul) sont connectées au bus de mise à jour (bus 32 bit synchrone).

Chaque cellule de calcul échange 64 bits de données avec les autres cellules à travers un réseau de FPGA d'échanges.

5. FPGA d'échanges

Les applications de traitement de signal demandent non seulement des puissances de calcul importantes mais aussi des capacités de transfert de données en proportion. C'est pourquoi nous avons consacré près de 40% des ressources de la carte aux échanges de données.

Nous avons implanté un réseau de cinq FPGA qui peuvent transférer simultanément 64 bits de chaque cellule vers une ou plusieurs cellules, chaque bit étant programmé individuellement. Ces FPGA d'échanges communiquent entre eux par des bus 32 bits. Au total, en tenant compte du bus de mise à jour, on peut transférer jusqu'à 3,2G octets par seconde entre les différents FPGA.

Ce réseau se charge également de connecter la carte avec l'extérieur à travers cinq liens de 12 bits chacun (un lien par FPGA d'échanges). La structure interne des FPGA permet de programmer chaque bit comme une entrée, une sortie ou une entrée – sortie, le protocole d'échange étant géré en interne par le FPGA en question. Quatre liens appelés liens internes sont destinés à des échanges avec d'autres cartes du PC, d'autres SYNTHUP par exemple. Un cinquième lien (lien externe) est relié au connecteur 37 points installé en bout de carte pour une connexion en dehors du PC. La capacité d'échange maximale est de 300M octets par seconde (horloge à 40 MHz).

Ces liens servent de supports pour échanger des données de synthèse (échantillons ou données intermédiaires) entre les cellules de calcul et l'extérieur. On peut aussi y transférer des informations de contrôle de synthèse grâce à la connexion au bus de mise à jour. Dans tous les cas, l'ordinateur hôte peut être récepteur ou destinataire de ces données.

6. Réalisation

Le circuit imprimé (carte « longue » au format PCI) est fabriqué selon une technologie standard de type 12 couches, classe 5. Les pistes d'horloge ont été particulièrement soignées afin de réduire les problèmes d'interférence. L'ensemble de la carte, excepté l'interface PCI, fonctionne sur une horloge commune à 40 MHz.

Nous avons adopté pour les FPGA un boîtier de type PQ240 qui permet de choisir les composants adaptés à la puissance de calcul requise pour l'application cible : On choisit des composants de la famille XL ou XLA (alimentation 3,3 V) qui s'échelonne du 4013 (équivalent à 10K-30K portes logiques) au 4085 (75K-200K portes logiques). De la même manière, la structure modulaire de la carte permet de n'installer qu'une partie des cellules, en fonction des besoins.

De même, le boîtier implanté pour les mémoires permettra de quadrupler la taille mémoire en passant des SDRAM 64 Mbits aux SDRAM 256 Mbits.

Pour les codes FPGA, nous utilisons la chaîne de compilation Xilinx standard : édition de schémas électriques sous Viewlogic ou édition de code VHDL, simulation, placement et routage avec contraintes temporelles.

7. Codes de synthèse

Les premiers codes en cours de développement sont destinés à l'UPIC, outil d'aide à la composition réalisé au CEMAMu.

SYNTHUP synthétise simultanément 800 oscillateurs échantillonnés à 48 kHz. Il s'agit de synthèse additive par lecture de formes d'ondes avec interpolation ; l'enveloppe et la fréquence sont interpolées à chaque échantillon ; chaque oscillateur peut être distribué sur quatre canaux.

Dans un premier temps, nous n'avons pas optimisé l'utilisation des ressources des FPGA. Une cellule gère une seule fonction : calcul de phase, interpolation de fréquence, lecture de forme d'onde avec interpolation, lecture d'enveloppe avec interpolation, cumul d'échantillons. La mise en forme des échantillons destinés à un convertisseur extérieur est réalisée par un FPGA d'échanges.

8. Driver Windows

Nous avons réalisé un driver de type VxD qui permet l'accès à la carte par programme sous Windows 95 et 98 (la prochaine version sera de type WDM, compatible en outre avec Windows 2000).

Mis en place de manière transparente dès le lancement du PC, il prend en compte les ressources allouées automatiquement à la carte par le gestionnaire *plug-and-play* de Windows 95 et 98. Un arbitrage évite les erreurs toujours possible lors d'une configuration manuelle, en particulier lors du partage des ressources du PC (dans notre cas, les plages d'adresses physiques de la mémoire et numéros d'interruption) ; l'utilisateur de notre carte n'aura donc aucune configuration à effectuer lors de l'installation.

Des points d'entrées permettent à un ou plusieurs programmes d'utiliser les services du VxD. On peut obtenir de cette façon, dans une application, des pointeurs sur les zones de mémoire internes de la carte : zone de code, de données et des registres de contrôle (lesquels sont en effet « mappés » en mémoire, afin d'en faciliter l'accès). La réservation des plages d'adresses virtuelles est faite au niveau du programme (permettant de choisir soit une plage d'adresses privées d'une application soit une plage commune à plusieurs applications) ; sur requête de l'application, le driver associe à ces adresses virtuelles, une plage d'adresses physiques dans la carte. L'accès à la mémoire de la carte n'étant pas paginé, on utilise cette mémoire comme n'importe quel bloc alloué par le système d'exploitation.

Le driver gère aussi les interruptions générées par le timer programmable de la carte ainsi que l'accès par la carte à la mémoire du PC (*bus-master DMA*).

9. Le bus PCI et le mode de transfert « *bus-master DMA* »

Le bus PCI, le plus rapide couramment disponible sur PC, possède un mode de fonctionnement particulièrement bien adapté à nos besoins : le mode de transfert « *bus-master DMA* », lequel permet à une carte électronique d'aller lire ou écrire elle-même la mémoire RAM du système, sans interruption et sans aucune intervention du processeur du PC. Une programmation préalable de la carte par le logiciel est également nécessaire pour qu'elle sache accéder aux données. Pour que cette technique soit pleinement efficace, nous devons l'employer non pas dans sa version simple, dans laquelle la mémoire lue doit être verrouillée (non

transférable (*swappable*) sur disque) et contiguë (inacceptable si les données occupent plusieurs méga-octets), mais en mode « *scatter-gather* ». Dans ce mode, on accède à la mémoire par un bloc contigu d'adresses virtuelles correspondant à des petits blocs de mémoire physique non nécessairement contigus (beaucoup plus efficace pour la gestion du système d'exploitation). Le logiciel doit donc charger dans la carte la table d'adressage qui lui permettra, pour chaque adresse virtuelle, de retrouver l'adresse physique correspondante (voir figure). Nous étudions une méthode pour ne verrouiller que les blocs physiques correspondant aux données servant à la synthèse et laisser le reste transférable (*swappable*) sur disque, toujours dans le but de conserver l'interactivité sans gêner le système d'exploitation.

10. Interface de programmation

La carte est accessible par une application de plusieurs manières : soit par l'utilisation directe des services du driver (programmation en C) soit à l'aide de la bibliothèque de classes C++ que nous fournissons avec la carte et qui facilite l'accès aux services du driver. Ces services sont évidemment disponibles quel que soit le code de synthèse chargé dans la carte.

D'autres outils, sous la forme de bibliothèque de fonctions et d'objets COM, sont en cours de développement ; ils sont destinés à faciliter le développement d'applications utilisant la synthèse additive (et donc le code de synthèse que nous utilisons nous-mêmes pour l'UPIC).

Deux programmes de test et de maintenance utilisant les services du VxD ont été réalisés : un programme de chargement de code dans la carte et un programme d'édition de la mémoire de la carte.

Chargement du code de synthèse : cette fonctionnalité, actuellement isolée dans un programme pour la simplicité du test, sera ultérieurement ajoutée au moniteur dont nous parlons ci-après. Ce programme permet de choisir sur le disque un fichier de code et de le télécharger dans la zone adéquate de la carte. Après cette opération, la carte devient pleinement opérationnelle. Chaque code détermine un mode de fonctionnement et des capacités particulières (de synthèse sonore par exemple) de la carte. Cette opération peut être répétée autant de fois que l'on veut, laissant ainsi la possibilité de changer de type de synthèse au cours d'une même session de travail. En outre, le chargement du code ne détruit pas les données qui peuvent ainsi, dans certains cas (codes ayant en commun certains formats de données) être utilisés par le nouveau code.

Moniteur : Cette application sert à la mise au point, aux tests et à la configuration de la carte de synthèse, elle affiche le contenu de ses mémoires et de ses registres de contrôle, et permet de les modifier. Capable d'accéder à la carte en même temps qu'une autre application, elle permet ainsi de vérifier le fonctionnement correct de cette dernière ; en particulier l'initialisation et l'évolution des paramètres internes de la carte.

11. Un exemple : la structure des données pour l'UPIC

Sans entrer dans des détails inutiles pour notre propos, on notera simplement que nous avons réparti les données entre la RAM du système et celle de la carte. Pour le code de synthèse, chaque type d'objet doit avoir une localisation déterminée à l'avance (dans la carte ou bien hors de la carte) ; en revanche, pour l'interface utilisateur, la localisation de chaque type d'objet peut être mixte : certains objets sont dans la carte, les autres en dehors. Cette dernière possibilité permet d'augmenter, si nécessaire, le nombre d'objets éditables (mais non utilisables pour la synthèse), ou bien de pouvoir faire de l'édition en l'absence de carte.

Voici, à titre d'illustration, quelques détails sur la structure des données qu'utilisent conjointement le code de synthèse et l'interface utilisateur de l'UPIC.

Localisation selon le type de donnée :

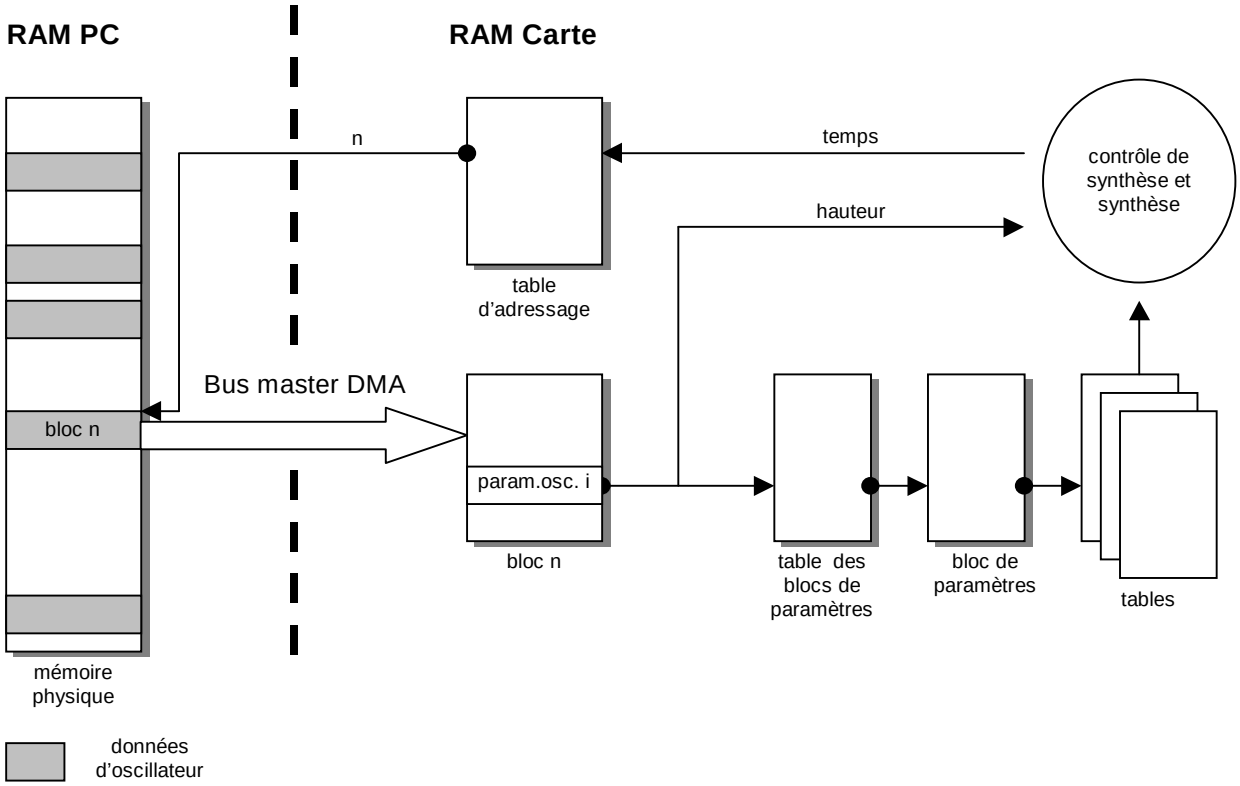
La carte utilise plusieurs types de données, auxquelles elle accède à des fréquences diverses. Par exemple :

- à chaque échantillon : formes d'onde
- toutes les millisecondes : enveloppes
- toutes les 10 millisecondes : données d'oscillateur

Pour des raisons d'optimisation, nous avons décidé d'inclure dans la carte la mémoire des formes d'onde, d'accès très fréquent et peu encombrante, et la mémoire des enveloppes. Nous mettrons à l'extérieur de la carte, la mémoire des données d'oscillateur, d'accès moins fréquent et constituant de loin le volume le plus important (elle peut atteindre plusieurs dizaines de Moctets).

Données d'oscillateur et tables d'adressage

Les données d'oscillateur se décomposent en une valeur de hauteur et un identificateur de bloc de paramètres, codés ensemble sur 32 bits. Cet identificateur, comme tous les identificateurs d'objets que nous utilisons est un index qui, entré dans une table d'adressage permet de retrouver l'adresse correspondante dans la RAM de la carte (voir figure). Cela nous permet d'une part de stocker sur disque des références à des objets indépendamment de leur emplacement en mémoire et d'autre part d'allouer dynamiquement la mémoire pour ces objets. Il y a ainsi dans la carte une table d'adressage par type d'objet stocké (bloc de paramètre, enveloppe, forme d'onde, tables de conversion).



SYNTHUP : données d'oscillateur et tables d'adressage

Les données de tous les oscillateurs pour un temps donné sont regroupées en un seul bloc. À chaque mise à jour, la carte lit dans la RAM du PC (et sans intervention du processeur du PC) les blocs de données dont elle a besoin. Leur localisation lui est fournie par une table d'adressage (temps / adresse physique) préalablement chargée par l'application.

12. Développements futurs

Nous nous proposons tout d'abord d'implémenter d'autres types de synthèse dans l'UPIC : modulation de fréquence, synthèse granulaire, etc. Constatant qu'une bonne partie des ressources de la carte reste disponible, nous allons explorer différentes méthodes d'analyse du signal, ce qui ouvre le champ des applications de type analyse – resynthèse. D'une façon plus générale, nous envisageons l'utilisation de notre carte dans des applications de type traitement de signal en temps réel. Le développement de ces applications sera facilité par la boîte à outil que nous réalisons en ce moment.

13. Bibliographie

PCI Local Bus Specification Rev. 2.2 protocol, electrical, mechanical and configuration specification for the PCI Local Bus components and expansion boards : http://pcisig.com/

Xilinx FPGAs: A Technical Overview for the First Time User v1.3, 12/98 : http://www.xilinx.com/xapp/xapp097.pdf

XC4000XL/XLA Series FPGAs : http://www.xilinx.com/products/xc4000XLA.html

Xilinx DSP Overview : http://www.xilinx.com/dsp/index.htm